Distributed Minimum Vertex Coloring and Maximum Independent Set in Chordal Graphs

Christian Konrad
Department of Computer Science, University of Bristol, UK
christian.konrad@bristol.ac.uk

Viktor Zamaraev
Department of Computer Science, Durham University, UK
viktor.zamaraev@durham.ac.uk

Abstract

We give deterministic distributed $(1+\varepsilon)$-approximation algorithms for Minimum Vertex Coloring and Maximum Independent Set on chordal graphs in the LOCAL model. Our coloring algorithm runs in $O(\frac{1}{\varepsilon}\log n)$ rounds, and our independent set algorithm has a runtime of $O(\frac{1}{\varepsilon}\log(\frac{1}{\varepsilon})\log^* n)$ rounds. For coloring, existing lower bounds imply that the dependencies on $\frac{1}{\varepsilon}$ and $\log n$ are best possible. For independent set, we prove that $\Omega(\frac{1}{\varepsilon})$ rounds are necessary.

Both our algorithms make use of the tree decomposition of the input chordal graph. They iteratively peel off interval subgraphs, which are identified via the tree decomposition of the input graph, thereby partitioning the vertex set into $O(\log n)$ layers. For coloring, each interval graph is colored independently, which results in various coloring conflicts between the layers. These conflicts are then resolved in a separate phase, using the particular structure of our partitioning. For independent set, only the first $O(\log \frac{1}{\varepsilon})$ layers are required as they already contain a large enough independent set. We develop a $(1+\varepsilon)$-approximation maximum independent set algorithm for interval graphs, which we then apply to those layers.

This work raises the question as to how useful tree decompositions are for distributed computing.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases local model, approximation algorithms, minimum vertex coloring, maximum independent set, chordal graphs

Digital Object Identifier 10.4230/LIPIcs.MFCS.2019.21

Related Version This work has previously been presented as a Brief Announcement at PODC 2018 [28]. A full version of the paper is available at [29], `https://arxiv.org/abs/1805.04544`.

Funding *Christian Konrad*: C.K. carried out most work on this paper while being at the University of Warwick. He was supported by the Centre for Discrete Mathematics and its Applications (DIMAP) at Warwick University and by EPSRC award EP/N011163/1.
Viktor Zamaraev: V.Z. is supported by EPSRC award EP/P020372/1.

1 Introduction

The LOCAL Model. In the LOCAL model of distributed computation [31], the input graph $G=(V,E)$ with $n=|V|$ represents a communication network, where every network node hosts a computational entity. Nodes have unique IDs. A distributed algorithm is executed on all network nodes simultaneously and proceeds in discrete rounds. Initially, besides their own IDs, nodes only know their neighbors. Each round consists of a computation and a communication phase. In the computation phase, nodes are allowed to perform unlimited computations. In the communication phase, nodes can send individual messages of unbounded sizes to all their neighbors (and receive messages from them as well). The runtime of the algorithm is the total number of communication rounds, and the objective is to design algorithms that run in as few rounds as possible. The output is typically distributed: For

vertex colorings, it is required that upon termination of the algorithm, every node knows its own color, and for independent sets, every node knows whether it participates in the independent set.

Network Decompositions. Network decompositions (see for example [3, 35, 32]) are a widely employed tool in distributed computing. They allow us to partition the vertex set of a graph into connected clusters of bounded diameter such that the cluster graph, i.e., the graph obtained by contracting the clusters into vertices, has small chromatic number. The Linial-Saks network decomposition method [32] guarantees that the cluster graph can be colored with $O(\log n)$ colors (and provides such a coloring), and no better method is known with regards to the number of colors. When employing a network decomposition algorithm for the Minimum Vertex Coloring (MVC) problem, the chromatic number of the cluster graph directly translates to the achieved approximation factor: Iterate through the color classes of the cluster graph and make each cluster color itself *optimally* using exponential time computations (recall that MVC is NP-hard) and using a color palette that is disjoint to the colors employed by already colored neighboring clusters. Indeed, this techniques yields the currently best known distributed algorithm for MVC (with approx. factor $O(\log n)$) [4].

Tree Decompositions. Our objective is to obtain distributed coloring algorithms with improved approximation guarantees, i.e., sub-logarithmic in n. To this end, we follow a route that at a first glance is very similar to the approach outlined above. However, instead of employing network decompositions, we make use of the *tree decomposition* of the input graph. A tree decomposition is a decomposition of the vertex set into (not necessarily disjoint) bags that are arranged in a tree such that every edge of the input graph is contained in the induced subgraph of at least one bag, and all bags that contain a specific vertex form a subtree in the tree decomposition. The key advantage of employing tree decompositions rather than network decompositions is that trees have small chromatic number and can efficiently be colored distributively. There are however multiple obstacles: First, there are graphs whose tree decompositions necessarily contain at least one bag whose diameter (maximum distance in the original graph between any two nodes in the bag) is $\Omega(n)$ (e.g. on a ring [16]). In these graphs, network nodes thus cannot learn the set of bags they are contained in in a small number of rounds and it appears impossible that nodes can obtain a local view of a coherent global tree decomposition. Second, the fact that every node may appear in multiple bags renders the coloring process more difficult since we cannot simply color independent parts of the tree simultaneously as some nodes may appear in multiple parts.

Our Results. Despite these obstacles, in this paper we show that the tree decomposition route can successfully be taken for the class of *chordal graphs*. A graph is chordal, if every cycle on at least four nodes contains a *chord*, i.e., an edge different from the edges of the cycle connecting two nodes of the cycle. Chordal graphs admit a tree decomposition where every bag has diameter 1, i.e., every bag is a clique. We first show that the tree decomposition of a chordal graph can efficiently be computed in the distributed setting. We then employ a peeling process on the tree decomposition that partitions the chordal graph into *interval subgraphs*, which can be colored efficiently distributively using a line of work by Halldórsson and Konrad [24, 25]. Particular care needs to be taken where these subgraphs meet, and we employ a color rotation technique to correct the colors at their boundaries. Perhaps surprisingly, this approach allows us to obtain a $(1+\varepsilon)$-approximation algorithm on this graph class, which constitutes our main result (**Theorem 14**). Our algorithm runs in $O(\frac{1}{\varepsilon}\log n)$ rounds and prior work shows that the dependencies on both $\frac{1}{\varepsilon}$ and $\log n$ are optimal.

We further adapt this approach to the Maximum Independent Set (MIS) problem. In general graphs, a $(1+\varepsilon)$-approximation to MIS can be obtained in $O(\frac{1}{\varepsilon}\log n)$ rounds [11, 21]. We show that using the tree decomposition approach outlined above, a $(1+\varepsilon)$-approximation on chordal graphs can be obtained in $O(\frac{1}{\varepsilon}\log(\frac{1}{\varepsilon})\log^* n)$ rounds (**Theorem 16**), and we prove that $\Omega(\frac{1}{\varepsilon})$ rounds are necessary (**Theorem 17**).

Related Work: Distributed Vertex Coloring. Distributed vertex coloring has been studied since more than 30 years (e.g. [14, 23]). Given a graph $G=(V,E)$, a *(legal) c-coloring* of G is an assignment $\gamma : V \to \{1,2,\ldots,c\}$ of at most c colors to the nodes of G such that every pair of adjacent nodes receives different colors. The algorithmic challenge lies in computing colorings with few colors. The *chromatic number* $\chi(G)$ is the smallest c such that there is a c-coloring. The MVC problem asks to find a $\chi(G)$-coloring. This is a difficult task, even in the centralized setting: In general graphs, MVC is NP-complete [27] and hard to approximate within a factor of $n^{1-\varepsilon}$, for any $\varepsilon > 0$ [39].

Most research papers on distributed vertex coloring address the problem of computing a $(\Delta+1)$-coloring (e.g. [14, 38, 37, 5, 8, 26, 18, 12]) (other objectives, such as Δ-colorings [36, 20] and other degree-based objectives [1] have been studied as well). Only few research papers address the MVC problem in a distributed model itself. On general graphs, the best distributed algorithm computes a $O(\log n)$-approximation in $O(\log^2 n)$ rounds [4] and is based on the network decomposition of Linial and Saks [32]. This algorithm uses exponential time computations, which due to the computational hardness of MVC is necessary unless $P=NP$. Barenboim et al. [7] gave a $O(n^{\varepsilon})$-approximation algorithm that runs in $\exp(O(1/\varepsilon))$ rounds. Both the exponential time computations and the relatively large best known approximation factor of $O(\log n)$ on general graphs motivate the study of special graph classes. Besides results on graph classes with bounded chromatic number (planar graphs [23] and graphs of bounded arboricity [6, 22]), the only natural graph class with unbounded chromatic number that has been addressed in the literature are *interval graphs*, which are the intersection graphs of intervals on the line. Halldórsson and Konrad gave a $(1+\varepsilon)$-approximation algorithm for MVC on interval graphs that runs in $O(\frac{1}{\varepsilon}\log^* n)$ rounds [25] (see also [24]). This work is the most relevant related work to our results.

Related Work: Distributed Independent Sets. An *independent set* in a graph $G=(V,E)$ is a subset of non-adjacent nodes $I \subseteq V$. Algorithms for independent sets are usually designed with one of the following two objectives in mind: (1) Compute a *maximal independent set*, i.e., an independent set I that cannot be enlarged by adding a node outside I to it, or (2) Compute a Maximum Independent Set (MIS) (or an approximation thereof), i.e., an independent set of maximum size, which is the variant studied in this paper. Similar to MVC, the MIS problem is NP-complete [27] and hard to approximate within a factor of $n^{1-\varepsilon}$, for every $\varepsilon > 0$ [39]. In the distributed setting, Luby [33] and independently Alon et al. [2] gave distributed $O(\log n)$ rounds maximal independent set algorithms more than 30 years ago. Improved results are possible for graphs with bounded maximum degree ([8, 19]) or on specific graph classes (e.g. [14, 38]). Using exponential time computations, a $(1+\varepsilon)$-approximation to MIS can be computed in general graphs in $O(\frac{1}{\varepsilon}\log n)$ rounds [11] (see also [21]). Deterministic distributed MIS algorithms may be inferior to randomized ones: It is known that every deterministic MIS $O(1)$-approximation algorithm on a path requires $\Omega(\log^* n)$ rounds [30, 15], while a simple randomized $O(1)$-round $O(1)$-approximation algorithm exists [15].

Outline. In Section 2, we give notation and definitions. We then discuss in Section 3 how network nodes can obtain coherent local views of the tree decomposition. A centralized $(1+\varepsilon)$-approximation MVC algorithm is then presented in Section 4, and distributed implementation of this algorithm is given in Section 5. Due to space restrictions, we only briefly sketch our results on MIS in Section 6 and defer a complete exposition to the full version of this paper [29]. Finally, we conclude in Section 7.

2 Preliminaries

Basic Notation and Definitions. Let $G = (V, E)$ be a graph. For a node $v \in V$, we denote by $\Gamma_G(v)$ the *neighborhood* of v in G. The *degree* of v in G is defined as $\deg_G(v) := |\Gamma_G(v)|$. By $\Gamma_G[v]$ we denote the set $\Gamma_G(v) \cup \{v\}$. Similarly, for a set of nodes $W \subseteq V$ we write $\Gamma_G(W) := (\bigcup_{v \in W} \Gamma_G(v)) \setminus W$, and $\Gamma_G[W] := \Gamma_G(W) \cup W$. The *distance-$k$ neighborhood* of v in G, i.e., the set of nodes at distance at most k from v in G, is denoted $\Gamma^k_G(v)$. The subgraph of G induced by a set of nodes U is denoted by $G[U]$. A set of pairwise adjacent (resp., non-adjacent) nodes in G is called a *clique* (resp., an *independent set*). A clique (resp., an independent set) S is *maximal* if $S \cup \{v\}$ is not a clique (resp., an independent set), for every $v \in V \setminus S$. A *maximum* independent set in G is an independent set of maximum size. The cardinality of a maximum independent set is called the *independence number* of G. A graph is *chordal* if every cycle of length at least four contains a chord, i.e., an edge that connects two non-consecutive nodes of the cycle. It is a well-known fact that an n-node chordal graph has at most n maximal cliques.

Tree Decomposition. A *tree decomposition* of an n-node graph $G = (V, E)$ is a forest $\mathcal{T} = (\mathcal{S}, \mathcal{E})$ whose vertex set $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ is a family of subsets of V, and:

1. every node[1] $v \in V$ belongs to at least one subset in $\mathcal{S}$;
2. for every edge $uv \in E$, there is a subset $S_i \in \mathcal{S}$ containing both nodes u and v;
3. for every node $v \in V$ the family $\phi(\mathcal{T}, v) \subseteq \mathcal{S}$ of subsets containing v induces a tree in $\mathcal{T}$, which we denote $\mathcal{T}(v)$, i.e., $\mathcal{T}(v) := \mathcal{T}[\phi(\mathcal{T}, v)]$.

When the tree decomposition is clear from the context we will write $\phi(v)$ instead of $\phi(\mathcal{T}, v)$. It follows from the definition that G is a subgraph of the intersection graph of the trees $\mathcal{T}(v)$.

Tree Decomposition of Chordal Graphs. It is well known (see, e.g., [10]) that a graph G is chordal if and only if it has a tree decomposition $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ whose vertex set $\mathcal{C}$ is the family of maximal cliques of G. We call such a tree decomposition a *clique forest* of chordal graph G. Since every vertex of the clique forest is a clique, G coincides with the intersection graph of the subtrees $\mathcal{T}(v)$ of clique forest $\mathcal{T}$. In other words, a clique forest of a chordal graph G is a forest $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ whose vertex set $\mathcal{C}$ is the family of maximal cliques of G, such that $\mathcal{T}[\phi(v)]$ is a tree for every v. If a clique forest of a chordal graph is linear, i.e., a forest with every component being a path, then the graph is interval.

▶ **Theorem 1** ([17]). *A chordal graph G is interval iff its clique forest is a linear forest.*

[1] For convenience, throughout the paper, we say *node* when referring to a vertex of an underlying graph, and we say *vertex* when referring to a vertex of its tree decomposition.

Binary Paths. We say that a path $v_1, \ldots, v_k$ in G is *binary*, if $\deg_G(v_i) \leq 2$, for every $i \in [k]$ (note that this implies that $\deg_G(v_i) = 2$, for every $i \in \{2, 3, \ldots, k-1\}$). We say that a binary path $v_1, \ldots, v_k$ is a *pendant path*, if either $\deg_G(v_1) = 1$ or $\deg_G(v_k) = 1$ (or both). For convenience, we consider an isolated vertex as a pendant path. A binary path $v_1, \ldots, v_k$ is an *internal path*, if $\deg_G(v_i) = 2$, for every $i \in [k]$. A binary/pendant/internal path is *maximal* if it cannot be enlarged by adding a vertex outside the path to it.

Let $G = (V, E)$ be a chordal graph with clique forest $\mathcal{T} = (\mathcal{C}, \mathcal{E})$. Let $\mathcal{P} = C_1, \ldots, C_k$ be a binary path in $\mathcal{T}$. We define the *diameter* of $\mathcal{P}$ to be the maximum distance in G between nodes in $C_1 \cup \ldots \cup C_k$, that is, $diam(\mathcal{P}) = \max_{u \in C_i, v \in C_j, i,j \in [k]} dist_G(u, v)$. Similarly, we define the *independence number* of $\mathcal{P}$ to be the independence number of $G[C_1 \cup \ldots \cup C_k]$.

Distributed Algorithms for Interval Graph. Halldórsson and Konrad [25] gave a deterministic distributed algorithm for coloring interval graphs. For every $\varepsilon \geq \frac{2}{\chi(G)}$, their algorithm computes a $(1+\varepsilon)$-approximation to MVC in $O(\frac{1}{\varepsilon} \log^* n)$ rounds. We will reuse this algorithm and denote it by ColIntGraph(ε).

3 Computing Local Views of the Clique Forest

Our algorithms make use of the clique forest of the input chordal graph. For network nodes to obtain a coherent view of the clique forest, we make use of the following *maximum weight spanning forest* characterization: With a chordal graph G we associate the *weighted clique intersection graph* $\mathcal{W}_G$ whose vertex set is the family $\mathcal{C}$ of maximal cliques of G, and any two cliques $C_1, C_2 \in \mathcal{C}$ with a nonempty intersection are connected by an edge with weight $|C_1 \cap C_2|$. Then:

▶ **Theorem 2** ([9]). *A forest $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ is a clique forest of a chordal graph G if and only if it is a maximum weight spanning forest of $\mathcal{W}_G$.*

Observe that while the vertex set of a clique forest is unique, i.e., the family of maximal cliques of G, the edge set is not necessarily unique as there may be multiple different maximum weight spanning forests in $\mathcal{W}_G$. To obtain coherent local views of a clique forest, it is thus necessary that nodes agree on a unique maximum weight spanning forest in $\mathcal{W}_G$. We achieve this by defining a linear order $<$ on the edges of $\mathcal{W}_G$ that respects the partial order given by the edge weights, and preferring edges that are larger with respect to $<$. To this end, we first assign to every maximal clique $C \in \mathcal{C}$ a word $\sigma(C)$ over the alphabet of the identifiers of nodes, where $\sigma(C)$ consists of the identifiers of the nodes in C listed in increasing order. Further, we associate with every edge $e = C_iC_j$ a triple (w_e, l_e, h_e), where w_e is the weight of e, i.e., $w_e = |C_i \cap C_j|$, $l_e = \text{lexmin}\{\sigma(C_i), \sigma(C_j)\}$, and $h_e = \text{lexmax}\{\sigma(C_i), \sigma(C_j)\}$. Now for two edges e and f we define $e < f$ if and only if either $w_e < w_f$, or $w_e = w_f$ and $l_e \prec l_f$, or $w_e = w_f$, $l_e = l_f$ and $h_e \prec h_f$, where $\prec$ is the lexicographical order. Clearly, $<$ orders the edges of $\mathcal{W}_G$ linearly while preserving the weight order.

In what follows, when we say that $\mathcal{T}$ is *the* clique forest of a chordal graph G, we implicitly assume that it is the clique forest uniquely specified by the above mechanism. Figure 2 demonstrates the weighted clique intersection graph and the clique forest of the chordal graph presented in Figure 1. A maximum weight spanning forest has the following easily verifiable local optimality property:

▶ **Lemma 3.** *Let $G = (V, E)$ be a weighted graph with a unique maximum weight spanning forest F, and let $U \subseteq V$ be a set of nodes inducing a tree T in F. Then $G[U]$ has a unique maximum weight spanning tree, which coincides with T.*

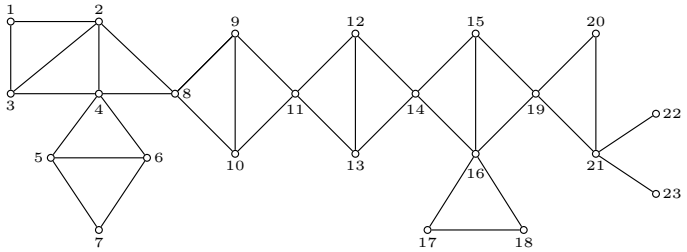

Figure 1 Chordal graph G.

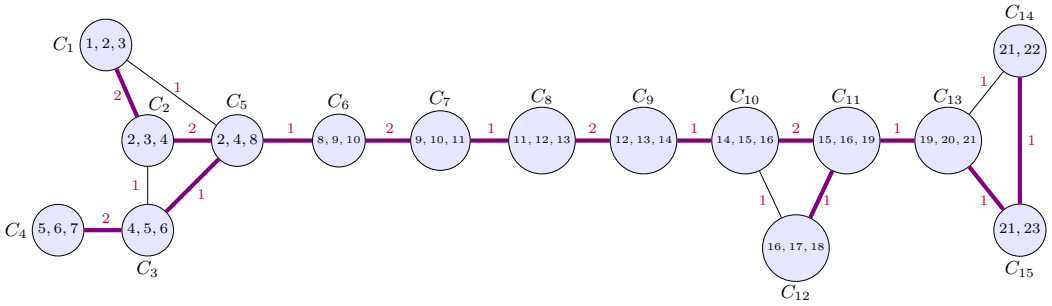

Figure 2 The weighted clique intersection graph $\mathcal{W}_G$ of chordal graph G presented in Fig. 1. The vertices of $\mathcal{W}_G$ are the maximal cliques of G, and two vertices C_i, C_j of $\mathcal{W}_G$ with a nonempty intersection are connected by an edge with weight $|C_1 \cap C_2|$. The bold edges are the edges of the clique forest $\mathcal{T}$ of G., i.e., the edges of the unique maximum weight spanning forest of $\mathcal{W}_G$ corresponding to the linear order of edges $<$.

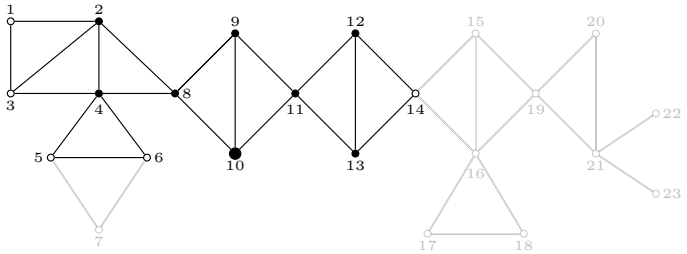

Figure 3 Local view of graph G from node 10. The non-gray nodes are the nodes in $\Gamma^3_G[10]$, and the black nodes are the nodes in $\Gamma^2_G[10]$.

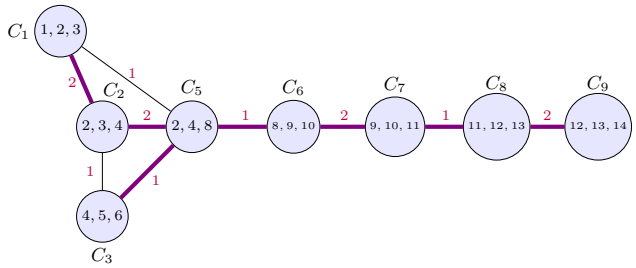

Figure 4 Local view of the graph $\mathcal{W}_G$ from node 10. The cliques in $\mathcal{C}' = \{C_1, C_2, C_3, C_5, C_6, C_7, C_8, C_9\}$ are exactly the maximal cliques of G that contain at least one node from $\Gamma^2_G[10]$. The bold edges are the edges of the unique maximum weight spanning forest of $\mathcal{W}_G[\mathcal{C}']$, which coincides with the subtree of $\mathcal{T}$ induced by $\mathcal{C}'$.

Applied to a chordal graph and its clique forest, we thus obtain:

▶ **Lemma 4.** *Let $G = (V, E)$ be a chordal graph and $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ its clique forest. Then for any $v \in V$ the unique maximum weight spanning forest in $\mathcal{W}_G[\phi(v)]$ equals to tree $\mathcal{T}(v) = \mathcal{T}[\phi(v)]$.*

This suggests a method for a node $v \in V$ to compute a local view $\mathcal{T}'$ of clique forest $\mathcal{T}$: Suppose that v knows its distance d-neighborhood $\Gamma^d_G[v]$. For every $u \in \Gamma^{d-1}_G[v]$, v computes the family $\phi(u)$ of maximal cliques containing u (notice that a maximal clique that contains a node at distance $d-1$ from v may include nodes at distance d). Then, v computes the maximum weight spanning forest in every $\mathcal{W}_G[\phi(u)]$ and adds the edges of this forest to $\mathcal{T}'$. Figures 3 and 4 illustrate construction of a local view of the clique forest of the chordal graph presented in Figure 1.

4 Minimum Vertex Coloring: Centralized Algorithm

In this section, we give a centralized $(1+\varepsilon)$-approximation algorithm for MVC on chordal graphs. This algorithm will later be implemented in the LOCAL model in Section 5.

4.1 Algorithm

Algorithm 1 A centralized $(1+\varepsilon)$-approximation coloring algorithm for chordal graphs.

Input: $G = (V, E)$ is an n-node chordal graph with clique forest $\mathcal{T} = (\mathcal{C}, \mathcal{E})$; a parameter $\varepsilon > \frac{2}{\chi(G)}$.

Set $k = 2/\varepsilon$.

(1) Pruning Phase.
Let $\mathcal{T}_1 = \mathcal{T}$, $U_1 = V$.
for $i = 1, 2, \ldots, \lceil \log n \rceil$ **do**:
- **a.** Let $\mathcal{L}_i$ be the set that contains all maximal pendant paths of $\mathcal{T}_i$, and all maximal internal paths of $\mathcal{T}_i$ of diameter at least $3k$.
- **b.** Let $V_i \subseteq U_i$ be such that for each $v \in V_i$, $\mathcal{T}(v)$ is a subpath of a path in $\mathcal{L}_i$.
- **c.** Let $U_{i+1} = U_i \setminus V_i$, and let $\mathcal{T}_{i+1}$ be the forest obtained from $\mathcal{T}_i$ by removing all paths in $\mathcal{L}_i$. As proved in Lemma 5, $\mathcal{T}_{i+1}$ is the clique forest of $G[U_{i+1}]$.

(2) Coloring Phase.
for $i = 1, 2, \ldots, \lceil \log n \rceil$ **do**: Color $G[V_i]$ with at most $(1 + 1/k)\chi(G[V_i]) + 1$ colors.

(3) Color Correction Phase.
for $i = \lceil \log n \rceil - 1, \lceil \log n \rceil - 2, \ldots, 1$ **do**:
for each path $\mathcal{P} \in \mathcal{L}_i$ **do**:
- **(a)** Let $W \subseteq V_i$ be the set of nodes w such that $\mathcal{T}(w)$ is a subpath of $\mathcal{P}$.
- **(b)** Let $W' \subseteq \bigcup_{l>i} V_l$ be the subset of nodes that have neighbors in W.
- **(c)** As we will show, $G[W \cup W']$ is an interval graph. Using Lemma 8, we recolor those nodes of W that are at distance at most $k+3$ from a node in W' using at most $(1 + 1/k)\chi(G[V_i]) + 1$ colors to resolve all coloring conflicts between W and W'.

Our algorithm (Alg. 1) consists of the pruning, the coloring, and the color correction phases: In the pruning phase, the node set V is partitioned into at most $\lceil \log n \rceil$ *layers* $V_1, \ldots, V_{\lceil \log n \rceil}$ such that, for every $i \in [\lceil \log n \rceil]$, $G[V_i]$ constitutes an interval graph. In each step of the pruning phase, we remove every node $v \in U_i$ from the current graph $G[U_i]$ (we set $U_1 = V$ and hence $G[U_1] = G$) whose corresponding subtree $\mathcal{T}(v)$ in the clique forest $\mathcal{T}_i$ of $G[U_i]$ is a subpath of a pendant path or an internal path of diameter at least $3k$. The set of removed nodes is denoted V_i, and $G[V_i]$ forms an interval graph (which follows from Lemma 7, [29]). We prove in Lemma 5 that the clique forest $\mathcal{T}_{i+1}$ of the resulting graph $G[U_{i+1}]$, where

$U_{i+1} = U_i \setminus V_i$, can be obtained by removing all pendant paths and all internal paths of diameter at least $3k$ from $\mathcal{T}_i$. We also show in Lemma 6 that the pruning process ends after at most $\lceil \log n \rceil$ iterations and thus creates at most $\lceil \log n \rceil$ layers.

In the coloring phase, each interval graph $G[V_i]$ is colored with at most $(1+1/k)\chi(G[V_i])+1$ colors. In the centralized setting, it would be easy to color these interval graphs optimally. However, since we will implement the algorithm later in the distributed setting, and an optimal coloring on interval graphs cannot be computed distributively in few rounds, we impose a weaker quality guarantee that can be achieved distributively. The colorings of different layers are computed independently from each other and do not give a coherent coloring of the entire input graph.

In the color correction phase, these incoherences are corrected. To this end, the colors of $V_{\lceil \log n \rceil}$ remain unchanged and we correct the layers iteratively, starting with layer $\lceil \log n \rceil - 1$ and proceeding downwards to layer V_1. In a general step, for every path $\mathcal{P} \in \mathcal{L}_i$, we show that the subgraph induced by the nodes $W \subseteq V_i$ whose subtrees are subpaths of $\mathcal{P}$ forms an interval graph together with those nodes in $\bigcup_{j \geq i+1} V_j$ that have coloring conflicts towards W (Lemma 8, [29]). Notice that each path $\mathcal{P}$ connects to at most two maximal cliques in $\mathcal{T}_i$. The neighborhood of W thus consists of subsets of these (at most two) cliques, which further implies that all conflicting nodes in $\bigcup_{j \geq i+1} V_j$ are included in these cliques as well. We then reuse a recoloring result previously proved by Halldórsson and Konrad [25], which shows that we can resolve all conflicts by changing the colors of those nodes in W that are at distance at most $k+3$ from the (at most) two conflicting cliques.

4.2 Analysis

The analysis of our algorithm relies on various technical lemmas that are given in the full version of this paper [29]. We now give the most important lemmas that allow us to prove correctness of our algorithm. Concerning the pruning step, we show that $\mathcal{T}_i$ is indeed the clique forest of $G[U_i]$ in Lemma 5, and we prove that at most $\lceil \log n \rceil$ iterations are required to complete the pruning process[2] in Lemma 6.

▶ **Lemma 5.** *For every i, $\mathcal{T}_i$ is the clique forest of $G[U_i]$.*

▶ **Lemma 6.** *Phase 1 in Alg. 1 requires at most $\lceil \log n \rceil$ iterations, i.e., $\bigcup_{1 \leq i \leq \lceil \log n \rceil} V_i = V$.*

Next, we address the color correction phase. In each iteration of the phase we consider every path $\mathcal{P} \in \mathcal{L}_i$ independently. The subgraph induced by the set of nodes $W \subseteq V_i$ whose corresponding trees are subpaths of $\mathcal{P}$ is legally colored in the coloring phase. This coloring may be inconsistent with the coloring of subgraph $G[U_{i+1}]$. However, we prove in Lemma 8, [29], that the set $W' \subseteq \bigcup_{s>i} V_s = U_{i+1}$ of neighbors of W in $G[U_{i+1}]$ (i.e., the nodes in U_{i+1} that could potentially cause conflicts) is the union of at most two cliques, which are included in the end vertices of the clique forest of interval graph $G[W \cup W']$. In order to resolve these conflicts we carry out a recoloring process on interval graph $G[W \cup W']$ with fixed colorings of its "boundary" cliques. To this end, we reuse a result by Halldórsson and Konrad [25]:

▶ **Lemma 7** (Halldórsson and Konrad [25])**.** *Let $G = (V, E)$ be an interval graph with its clique forest $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ being a path $\mathcal{P} = C_1, C_2, \ldots, C_k$ such that $\mathrm{dist}_G(u, v) \geq r$ for every pair of nodes $u \in C_1, v \in C_k$, for an integer $r \geq 5$. Suppose that cliques C_1 and C_k are legally colored using at most c colors. Then the coloring of $G[C_1 \cup C_k]$ can be extended to a legal coloring of G with at most $\max\{\lfloor (1 + \frac{1}{r-3})\chi(G) \rfloor + 1, c\}$ colors.*

[2] Recently, inspired by Miller and Reif's *parallel tree contraction* [34], a very similar procedure was developed by Chang and Pettie [13].

Equipped with Lemma 7, we now prove correctness of the color correction phase.

▶ **Lemma 8** (Recoloring Lemma). *Consider the color correction phase (Step 3) of Algorithm 1. Let $\mathcal{P} \in \mathcal{L}_i$ be a path and let $W \subseteq V_i$ be the subset of nodes whose corresponding subtrees are included in $\mathcal{P}$. Further, let $W' \subseteq \bigcup_{s>i} V_s = U_{i+1}$ be the nodes in U_{i+1} that have neighbors in W. Suppose that W' is colored using colors from the set $[\lfloor(1+1/k)\chi(G)+1\rfloor]$. Then, we can recolor those nodes of W that are at distance at most $k+4$ from W' in G with colors from the set $[\lfloor(1+1/k)\chi(G)+1\rfloor]$ so that $G[W \cup W']$ is legally colored.*

Proof. By Lemma 8, [29], $G[W \cup W']$ is an interval graph and its clique forest is a path. Let $\mathcal{P}' = C_1, C_2, \ldots, C_r$ denote this path. The same lemma also states that $W' \subseteq C_1 \cup C_r$.

Let i be the minimum index such that $\operatorname{dist}(u,v) \geq k+3$, for every $u \in W' \cap C_1$ and $v \in C_i$. Then, by Lemma 7, the nodes of the cliques $C_2, \ldots, C_{i-1}$ can be recolored using at most $\lfloor(1+1/k)\chi(G)\rfloor+1$ colors to resolve the coloring conflicts between $W' \cap C_1$ and W. Similarly, let j be the maximum index such that $\operatorname{dist}(u,v) \geq k+3$, for every $u \in W' \cap C_r$ and $v \in C_j$. Then, by Lemma 7, the nodes of the cliques $C_{j+1}, \ldots, C_{r-1}$ can be recolored using at most $\lfloor(1+1/k)\chi(G)\rfloor+1$ colors to resolve the conflicts between $W' \cap C_r$ and W. ◀

▶ **Theorem 9.** *For any $\varepsilon > \frac{2}{\chi(G)}$, Algorithm 1 is a $(1+\varepsilon)$-approximation* MVC *algorithm on chordal graphs.*

Proof. First, we show by induction that the algorithm uses at most $(1+1/k)\chi(G)+1$ colors. This is clearly true for $G_{\lceil \log n \rceil}$. The induction step follows from Lemma 8. Now, using the assumption $\varepsilon > \frac{2}{\chi(G)}$, we obtain: $(1+1/k)\chi(G)+1 \leq (1+\varepsilon/2)\chi(G)+\varepsilon\chi(G)/2 = (1+\varepsilon)\chi(G)$, which proves the approximation factor of the algorithm. ◀

5 Minimum Vertex Coloring: Distributed Algorithm

We give now a LOCAL model implementation of Algorithm 1 that runs in $\mathrm{O}\left(\frac{1}{\varepsilon}\log n\right)$ rounds.

5.1 Algorithm

The global behavior of our distributed algorithm, Algorithm 2, is identical to that of our centralized Algorithm 1. The main challenge lies in the coordination of the network nodes. One particular difficulty stems from the fact that network nodes are not aware of n, the total number of nodes, and thus do not know when the $\lceil \log n \rceil$ iterations of the pruning phase have completed. For this reason, nodes execute the three phases of Algorithm 1 asynchronously.

We will first present the pseudocode of our distributed algorithm, which is executed independently on every node v. Then we will describe each of the three phases in detail.

Algorithm 2 A distributed $(1+\varepsilon)$-approximation algorithm, code for node v.

Input: a parameter ε, let $k = \lceil 2/\varepsilon \rceil$

1. **Pruning Phase.** $(l_v, parent_v, children_v) \leftarrow$ PRUNETREE()
2. **Coloring Phase.** Run COLINTGRAPH($\frac{1}{k}$) on layer l_v and store color in c_v
3. **Color Correction Phase.**
   if $parent_v \neq \perp$ **then**
   Wait until message SETCOLOR(c) received from $parent_v$; Set $c_v \leftarrow c$;
   end if
   CORRECTCHILDREN($children_v, k$)

The Pruning Phase. In the pruning phase, the subroutine PruneTree is invoked and returns parameters l_v, $parent_v$, and $children_v$, where l_v is the layer of node v, and $parent_v$ and $children_v$ are variables necessary for the coordination of the color correction phase and are defined and explained later. The pseudocode of PruneTree is given in Algorithm 3.

Algorithm 3 PruneTree(), code for node v.

Initialization:
Let $i = 1$, $l_v = -1, children_v = \{\}$, and $parent_v = \perp$

while $l_v = -1$ **do**:

1. Collect $\Gamma_G^{10k}(v)$ together with variables l_u and ID_u, for every $u \in \Gamma_G^{10k}(v)$
2. Compute local view of the clique forest $\mathcal{T}_i = (\mathcal{C}_i, \mathcal{E}_i)$ of the subgraph of G induced by the nodes $u \in \Gamma_G^{10k}(v)$ with $l_u = -1$
3. **if** $\mathcal{T}_i(v)$ is a subpath of a pendant path in $\mathcal{T}_i$, or $\mathcal{T}_i(v)$ is a subpath of a binary path in $\mathcal{T}_i$ of diameter at least $3k$ **then**
   $l_v = i$; $parent_v =$ parent of v;
   else
   Add children in layer i (if there are any) to $children_v[i]$
   end if
4. $i = i + 1$

return $(l_v, parent_v, children_v)$

In each iteration of the while loop of PruneTree, one layer is removed from the clique forest of the input graph. To describe the global behavior of the algorithm, we will reuse the naming conventions already used in Algorithm 1. Let $U_1 = V$, and let $V_i \subseteq U_i$ be the set of nodes removed in iteration i, i.e., assigned layer index i. Let also $U_{i+1} = U_i \setminus V_i$, and let $\mathcal{L}_i$ be the set of maximal paths removed from the clique forest $\mathcal{T}_i$ of $G[U_i]$.

In each iteration i, first, each node v collects its distance-$10k$ neighborhood. Then, v computes its local view of the clique forest $\mathcal{T}_i$ of the graph induced by the nodes that have not yet been removed from the graph, i.e., of $G[U_i]$ (as in Section 3). Next, node v is removed from $G[U_i]$ and added to the current layer V_i if its corresponding subtrees $\mathcal{T}_i(v)$ is entirely contained in either a pendant path or a binary path of large enough diameter. This step is identical to Algorithm 1, and the exact same partitioning is computed. Node v that is removed in the current iteration stores its parent in $parent_v$, and nodes that remain in the graph potentially store some of the removed nodes as their children in $children_v$.

▶ **Definition 10** (Parent, Child). *Let $v \in V_i$ and let $\mathcal{P}$ be the maximal binary path in $\mathcal{T}_i$ that contains $\mathcal{T}_i(v)$. If $\mathcal{P}$ is a component of $\mathcal{T}_i$ then we define $parent_v := \perp$. Otherwise, let C be the vertex outside $\mathcal{P}$ in $\mathcal{T}_i$ such that C is adjacent to an end vertex of $\mathcal{P}$ and $dist_G(v, C)$ is minimal. Let c be the node with maximum ID in C. Then the* parent *of v is defined to be c, if $dist_G(v, C) \leq k + 3$, and $\perp$ otherwise. If c is the parent of v, then v is a* child *of c.*

The parent of node v is responsible for recoloring v in the color correction phase. Notice that a node v does not have a parent if the closest maximal clique outside v's path $\mathcal{P}$ is at least at distance $k + 4$ from v. In this case, the color that v will receive in the coloring phase is final and no color correction is needed for v. Recall that in the color correction phase of Algorithm 1, we only need to recolor nodes that are at distance at most $k + 3$ from the cliques that contain nodes with color conflicts. Finally, the subroutine returns the node's level l_v, its parent $parent_v$, and its children $children_v$.

The Coloring Phase. Notice that all nodes of layer i return from PruneTree in the same round. They can hence invoke the coloring phase simultaneously. They run the algorithm ColIntGraph of Halldórsson and Konrad [25] and compute a coloring on $G[V_i]$ that uses at most $\lfloor(1+\frac{1}{k})\chi(G[V_i])+1\rfloor$ colors. This algorithm runs in $O(k \log^* n)$ rounds.

While some nodes execute the coloring phase, others still execute PruneTree. These nodes repeatedly collect their distance-$10k$ neighborhood. This requires all other network nodes to continuously forward messages, which can be taken care of in the background.

The Color Correction Phase. In color correction phase, nodes with assigned parents (i.e., nodes v with $parent_v \neq \perp$) first wait until they received their final color from their parents. Only then they proceed and correct the colors of their children. To this end, each such node v runs subroutine CorrectChildren, which processes $children_v$ layer by layer, starting with layer $l_v - 1$ down to 1. If v has children in layer V_i, then it waits until all nodes adjacent to $children_v[i]$ which are contained in layers $> i$ have received their final colors. This can be done by repeatedly collecting its local distance-$(k+5)$ neighborhood and checking whether the colors of all nodes in $\Gamma_{G[U_i]}(children_v[i])$ are final. Then, v locally computes the color correction for $children_v[i]$ and notifies them about their new colors.

Algorithm 4 CorrectChildren($children_v, k$), code for node v.

for $l \leftarrow l_v - 1, l_v - 2, \ldots, 1$ **do:**
  if $children_v[l] \neq \{\}$ **then**
    a. Wait until all neighbors of $children_v[l]$ in $G[U_l]$ have received their final color
    b. Compute color correction as in Lemma 8
    c. For each $u \in children_v[l]$, send message SetColor(c) to u, where c is u's new color

5.2 Analysis

To ensure correctness of our algorithm, we need to show that the parent of a node $v \in V_i$ is contained in a layer $j > i$. This is shown via the following lemma.

▶ **Lemma 11.** *Let $\mathcal{P} \in \mathcal{L}_i$, and let $W \subseteq V_i$ be the set of nodes whose corresponding subtrees are included in $\mathcal{P}$. Then every node $u \in \Gamma_{G[U_i]}(W)$ is contained in a layer V_j with $j > i$.*

It follows from the definition that the parent of a node $v \in V_i$ belongs to $\Gamma_{G[U_i]}(W)$. Hence:

▶ **Corollary 12.** *The parent of a node $v \in V_i$ is contained in a layer V_j with $j > i$.*

The following lemma demonstrates that Algorithm 2 mimics the behavior of our centralized algorithm and uses $O(\frac{1}{\varepsilon}\log n)$ rounds. This establishes our main result, stated in Theorem 14.

▶ **Lemma 13.** *The global behavior of Algorithm 2 is identical to the behavior of Algorithm 1. Furthermore, Algorithm 2 runs in $O(\frac{1}{\varepsilon}\log n)$ rounds.*

▶ **Theorem 14.** *For every $\varepsilon \geq \frac{2}{\chi(G)}$, there is a deterministic $(1+\varepsilon)$-approximation algorithm for MVC on chordal graphs that runs in $O(\frac{1}{\varepsilon}\log n)$ rounds in the LOCAL model.*

6 Maximum Independent Set

Maximum Independent Set on Interval Graphs. Let $H = (V, E)$ be an interval graph, i.e., a chordal graph whose clique forest is a collection of paths. We first observe that the subset of nodes $V' \subseteq V$, with $u \in V'$ iff there exists a node $v \in V$ with $\Gamma_H[v] \subsetneq \Gamma_H[u]$, is not needed

for the computation of a large maximum independent set: If a maximum independent set I^* in H contains a node $u \in V'$, then it can simply be replaced by a node v with $\Gamma_H[v] \subsetneq \Gamma_H[u]$. We thus only need to consider graph $H' := H[V \setminus V']$, which constitutes a unit interval graph. We first compute a distance-k maximal independent set I in $O(k \log^* n)$ rounds, for some $k = \Theta(\frac{1}{\varepsilon})$, by simulating the maximal independent set $O(\log^* n)$ rounds algorithm for bounded-independence graphs [38] on H'^k. Then, every $(v_1, v_2) \in P$, where P is the set of pairs of nodes of I that are of mutual distance at most $2k-1$, computes a maximum independent set I_{v_1,v_2} among the nodes *located between them* but outside of $\Gamma_{H'}(v_1) \cup \Gamma_{H'}(v_2)$ in $O(k)$ rounds. Set $I \cup (\cup_{(v_1,v_2)\in P} I_{v_1,v_2})$ has the desired size.

▶ **Theorem 15.** *For every $\varepsilon > 0$, there is a deterministic $(1+\varepsilon)$-approximation algorithm for* MIS *on interval graphs that operates in $O(\frac{1}{\varepsilon} \log^* n)$ rounds in the LOCAL model.*

Maximum Independent Set on Chordal Graphs. Our distributed MIS algorithm uses an adapted version of the peeling process used in our coloring algorithm. The key observation that allows us to obtain a runtime of $o(\log n)$ is the fact that the first $O(\log \frac{1}{\varepsilon})$ layers computed by our peeling process already contain a large enough independent set. Our algorithm proceeds as follows: In each iteration $i = 1, \ldots, O(\frac{1}{\varepsilon})$ of the peeling process, we remove set $\mathcal{L}_i$ of all pendant paths and all internal paths of large enough diameter from the clique forest $\mathcal{T}_i$ of the graph induced by the remaining nodes. Next, we compute large independent sets among the nodes whose trees are included in each path $\mathcal{P} \in \mathcal{L}_i$. If $\mathcal{P}$ has a large independence number then we run our $(1+\varepsilon)$-approximation algorithm for interval graphs in $O(\frac{1}{\varepsilon} \log^* n)$ rounds. If $\mathcal{P}$ has small independence number we need to compute an optimal independent set in order to locally stay within a $(1+\varepsilon)$-approximation guarantee. This can be achieved using only $O(\frac{1}{\varepsilon})$ rounds, since paths with small independence number necessarily have small diameter. The runtime is dominated by the product of the number of iterations $O(\log \frac{1}{\varepsilon})$ and the $O(\frac{1}{\varepsilon} \log^* n)$ runtime of our MIS algorithm for interval graphs.

▶ **Theorem 16.** *For any $\varepsilon \in (0, 1/2)$, there is a deterministic $(1+\varepsilon)$-approximation algorithm for* MIS *on chordal graphs that runs in $O(\frac{1}{\varepsilon} \log(\frac{1}{\varepsilon}) \log^* n)$ rounds in the LOCAL model.*

When implementing this idea, care needs to be taken when combining the computed independent sets of different levels. Indeed, our algorithm bases the computation of the independent set in level i on the outcome of the computations of the independent sets of levels $< i$ and it seems difficult to avoid this. We therefore cannot execute the independent set computations of different levels simultaneously, which would reduce the runtime to $O(\frac{1}{\varepsilon}(\log^* n + \log \frac{1}{\varepsilon}))$.

Lower Bound. Our lower bound is established on a path P_n of length n and uses an indistinguishability argument: Consider a k rounds MIS algorithm that operates on P_n, and let v be an arbitrary node that is not too close to an end point of the path. Suppose that v is selected into the independent set. Let u_1 be a node at distance $2k+1$ from v and let u_2 be the adjacent node to u_1 that is at distance $2k+2$ from v. Since the runtime of the algorithm is k, the outputs computed by u_1 and u_2 are independent from v. Since in average it is equally likely that u_1 or u_2 are selected, the expected size of an independent set in the subpath starting at v and ending at u_2 is thus strictly smaller than $k+2$, while a maximum independent set in this subpath is of size $k+2$. Based on this insight, we obtain

▶ **Theorem 17.** *For every $\varepsilon > 0$ and n large enough, every randomized algorithm in the LOCAL model with expected approximation factor at most $1+\varepsilon$ for* MIS *requires $\Omega(\frac{1}{\varepsilon})$ rounds.*

7 Conclusion

In this paper, we gave distributed $(1+\varepsilon)$-approximation algorithms for MVC and MIS on chordal graphs. We showed that in chordal graphs network nodes can obtain coherent views of a global tree decomposition, which enabled us to exploit the tree structure of the input graph for the design of algorithms. How can we extend the class of graphs on which we can solve MVC and MIS within a small approximation factor even further? In particular, how can we handle graphs that contain longer induced cycles, such as k-chordal graphs (for some integer k)?

References

1 Pierre Aboulker, Marthe Bonamy, Nicolas Bousquet, and Louis Esperet. Distributed Coloring in Sparse Graphs with Fewer Colors. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, PODC '18, pages 419–425, New York, NY, USA, 2018. ACM. doi:10.1145/3212734.3212740.

2 Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986. doi:10.1016/0196-6774(86)90019-2.

3 Baruch Awerbuch, Michael Luby, Andrew V Goldberg, and Serge A Plotkin. Network Decomposition and Locality in Distributed Computation. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, pages 364–369, Washington, DC, USA, 1989. IEEE Computer Society. doi:10.1109/SFCS.1989.63504.

4 Leonid Barenboim. On the Locality of Some NP-complete Problems. In *Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part II*, ICALP'12, pages 403–415, Berlin, Heidelberg, 2012. Springer-Verlag. doi:10.1007/978-3-642-31585-5_37.

5 Leonid Barenboim. Deterministic $(\Delta+1)$-Coloring in Sublinear (in Δ) Time in Static, Dynamic and Faulty Networks. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC '15, pages 345–354, New York, NY, USA, 2015. ACM. doi:10.1145/2767386.2767410.

6 Leonid Barenboim and Michael Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. *Distributed Computing*, 22(5):363–379, 2010.

7 Leonid Barenboim, Michael Elkin, and Cyril Gavoille. A Fast Network-Decomposition Algorithm and Its Applications to Constant-Time Distributed Computation. In *Post-Proceedings of the 22nd International Colloquium on Structural Information and Communication Complexity - Volume 9439*, SIROCCO 2015, pages 209–223, New York, NY, USA, 2015. Springer-Verlag New York, Inc. doi:10.1007/978-3-319-25258-2_15.

8 Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The Locality of Distributed Symmetry Breaking. *J. ACM*, 63(3):20:1–20:45, June 2016. doi:10.1145/2903137.

9 Philip A Bernstein and Nathan Goodman. Power of natural semijoins. *SIAM Journal on Computing*, 10(4):751–771, 1981.

10 Jean RS Blair and Barry Peyton. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*, pages 1–29. Springer, 1993.

11 Marijke H.L. Bodlaender, Magnús M. Halldórsson, Christian Konrad, and Fabian Kuhn. Brief Announcement: Local Independent Set Approximation. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, PODC '16, pages 93–95, New York, NY, USA, 2016. ACM. doi:10.1145/2933057.2933068.

12 Yi-Jun Chang, Wenzheng Li, and Seth Pettie. An optimal distributed $(\Delta+1)$-coloring algorithm? In *Proceedings 50th ACM Symposium on Theory of Computing (STOC)*, pages 445–456, 2018.

13 Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the LOCAL model. *SIAM Journal on Computing*, 48(1):33–69, 2019.

14 Richard Cole and Uzi Vishkin. Deterministic Coin Tossing with Applications to Optimal Parallel List Ranking. *Inf. Control*, 70(1):32–53, July 1986. doi:10.1016/S0019-9958(86)80023-7.

15 Andrzej Czygrinow, Michal Hańćkowiak, and Wojciech Wawrzyniak. Fast Distributed Approximations in Planar Graphs. In Gadi Taubenfeld, editor, *Distributed Computing*, pages 78–92, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

16 Yon Dourisboure and Cyril Gavoille. Tree-decompositions with Bags of Small Diameter. *Discrete Math.*, 307(16):2008–2029, July 2007. doi:10.1016/j.disc.2005.12.060.

17 Peter C Fishburn. *Interval orders and interval graphs: A study of partially ordered sets*. John Wiley & Sons, 1985.

18 Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local Conflict Coloring. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 625–634, 2016.

19 Mohsen Ghaffari. An Improved Distributed Algorithm for Maximal Independent Set. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 270–277, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2884435.2884455.

20 Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, and Yannic Maus. Improved Distributed Delta-Coloring. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, PODC '18, pages 427–436, New York, NY, USA, 2018. ACM. doi:10.1145/3212734.3212764.

21 Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the Complexity of Local Distributed Graph Problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 784–797, New York, NY, USA, 2017. ACM. doi:10.1145/3055399.3055471.

22 Mohsen Ghaffari and Christina Lymouri. Simple and Near-Optimal Distributed Coloring for Sparse Graphs. In *Distributed Computing: 31th International Symposium, DISC 2017, Vienna, Austria, October 16-20, 2017. Proceedings*, 2017.

23 Andrew V. Goldberg, Serge A. Plotkin, and Gregory E. Shannon. Parallel Symmetry-Breaking in Sparse Graphs. *SIAM J. Discrete Math.*, 1(4):434–446, 1988. doi:10.1137/0401044.

24 Magnús M. Halldórsson and Christian Konrad. *Distributed Algorithms for Coloring Interval Graphs*, pages 454–468. Springer, 2014. doi:10.1007/978-3-662-45174-8_31.

25 Magnús M. Halldórsson and Christian Konrad. Improved Distributed Algorithms for Coloring Interval Graphs with Application to Multicoloring Trees. In *Post-Proceedings of the 24th International Colloquium on Structural Information and Communication Complexity*, SIROCCO 2017, 2017.

26 David G. Harris, Johannes Schneider, and Hsin-Hao Su. Distributed $(\Delta + 1)$-coloring in Sublogarithmic Rounds. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 465–478, New York, NY, USA, 2016. ACM. doi:10.1145/2897518.2897533.

27 Richard M Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

28 Christian Konrad and Viktor Zamaraev. Brief Announcement: Distributed Minimum Vertex Coloring and Maximum Independent Set in Chordal Graphs. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 159–161, 2018. doi:10.1145/3212734.3212787.

29 Christian Konrad and Viktor Zamaraev. Distributed Coloring and Independent Set in Chordal Graphs. *arXiv preprint arXiv:1805.04544.*, 2018.

30 Christoph Lenzen and Roger Wattenhofer. Leveraging Linial's Locality Limit. In Gadi Taubenfeld, editor, *Distributed Computing*, pages 394–407, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

31 Nathan Linial. Locality in Distributed Graph Algorithms. *SIAM J. Comput.*, 21(1):193–201, February 1992. doi:10.1137/0221015.

32 Nathan Linial and Michael Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, December 1993. doi:10.1007/BF01303516.

33 Michael Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986. doi:10.1137/0215074.

34 Gary L. Miller and John H. Reif. Parallel Tree Contraction–Part I: Fundamentals. In *Randomness and Computation*, volume 5, pages 47–72. JAI Press, 1989.

35 Alessandro Panconesi and Aravind Srinivasan. Improved Distributed Algorithms for Coloring and Network Decomposition Problems. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 581–592, New York, NY, USA, 1992. ACM. doi:10.1145/129712.129769.

36 Alessandro Panconesi and Aravind Srinivasan. The Local Natur of Delta-Coloring and its Algorithmic Applications. *Combinatorica*, 15(2):255–280, 1995. doi:10.1007/BF01200759.

37 Johannes Schneider and Roger Wattenhofer. A New Technique for Distributed Symmetry Breaking. In *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '10, pages 257–266, New York, NY, USA, 2010. ACM. doi:10.1145/1835698.1835760.

38 Johannes Schneider and Roger Wattenhofer. An optimal maximal independent set algorithm for bounded-independence graphs. *Distributed Computing*, 22(5):349–361, August 2010. doi:10.1007/s00446-010-0097-1.

39 David Zuckerman. Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. *Theory of Computing*, 3(1):103–128, 2007.